R-C Network Measurement with BASCOM AVR

Better accuracy from a new command

By J. C. Feltes

The popular BASIC compiler for AVR microcontrollers has been expanded with a very useful command. Instead of using the inaccurate GetRC0 command, you're now able to establish R-C values by means of two port pins.

When a microcontroller has no analogue/digital converter, you need to resort to tricks if you want to measure the value of a resistor or a capacitor in an R-C network. For this purpose, BASCOM AVR has the command GetRC available. With reference to **Figure 1**, the R-C network connected to port line D4 should have a 100-nF capacitor while the value of the resistor to be measured is between 0 and 10 kΩ.

GetRC arranges for the capacitor to be charged in one go, and discharged in small steps until the Low state is reached. The number steps are counted and stored in a 'word' variable (**Listing 1**).

Of course, the display should be prepared and configured to enable the micro to show the measured value. Unfortunately, sound as the above theoretical concept may appear, it fails miserably in practice. For example, two successive measurements (10 kΩ and 4.7 kΩ) using the fixed 100-nF capacitor produced the following values on the display: 756 and 508 respectively.

In order to improve the accuracy an experiment could be set up in which the GetRC command is mimicked by a series of discrete steps. A suggested software tool is shown in **listing 2**. In the first part, the capacitor is charged during 50 ms to make sure it is 'full'. Next, the relevant port pin is declared an 'output' and pulled Low. Each of the commands takes 0.5 μs to complete (assuming a 4-MHz clock used on an AT90S8535 which requires two machine cycles per instruction). Next, the port pin is configured as an input again and the software runs a check to see if the capacitor voltage has dropped Low already. Switching from output to input again takes 0.5 μs. If the capacitor is not yet discharged, the variable is incremented (0.4 μs) before the return occurs.

Listing 2 shows different resistor values entered for GetRC and $$GetRC. Clearly, the measurement principle only yields reasonably reliable results for resistor values up to about 10 kΩ. At higher resistance values, the measured values even seem to drop (probably because of various leakage currents). However,

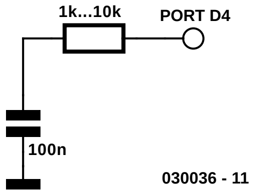


Figure 1. Circuit configuration for the original GetRC command and $$getrc.bas.

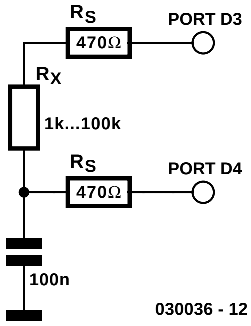
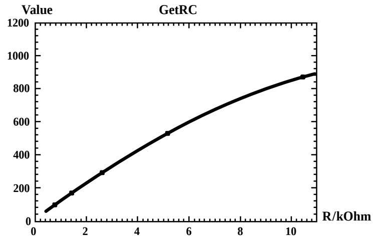


Figure 2. Two port pins are needed for a precision measurement using $$getrc2.bas.

Listing 1: measurement using original command GetRC

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.b ,
                Db7 = Portc.7 , Rs = Portd.7 , E = Portd.6
Config Lcd = 20 * 2
Dim W As Word
Do
   W = Getrc(pind , 4)
   Cls
   Lcd W
Loop
```

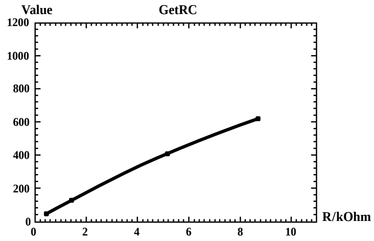
030036 - 14



Listing 2: measurement using $$getrc.bas according to MSC method

```
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.b ,
                Db7 = Portc.7 , Rs = Portd.7 , E = Portd.6
Config Lcd = 20 * 2
Dim W As Word

Do
   Config Pind.4 = Output
   Portd.4 = 1
   Waitms 50
   W = 0
   Do
      Config Pind.4 = Output
      Portd.4 = 0
      Config Pind.4 = Input
      If Pind.4 = 0 Then Exit Do
      Else
         W = W + 1
         If W > 65000 Then Exit Do
      End If
   Loop
   Cls
   Lcd W
Loop
```



030036 - 15

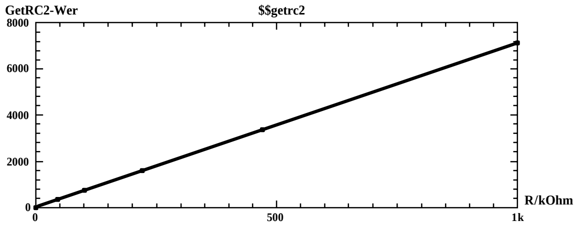
Listing 3: measurement using $$getrc2.bas for 2 port pins

```
Declare Sub Getrc2(w As Word)
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.b ,
                Db7 = Portc.7 , Rs = Portd.7 , E = Portd.6
Config Lcd = 20 * 2
Dim W As Word

Do
   Call Getrc2(w)
   Cls
   Lcd W
Loop

Sub Getrc2(w)
Config Pind.4 = Output
Config Pind.3 = Output

   Portd.4 = 1
   Portd.3 = 1
   Waitms 10
   W = 0
   Config Pind.4 = Input
   Portd.4 = 0
   Portd.3 = 0
   Do
      If Pind.4 = 0 Then Exit Do
      Else
         W = W + 1
         If W > 65000 Then Exit Do
      End If
   Loop
   Config Pind.4 = Output
   Portd.4 = 1
   Portd.3 = 1
End Sub
```



030036 - 16

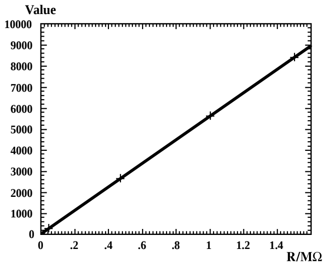
Listing 4: measurement using $$getrc3.bas (after Jos) on one port pin

```
Declare Sub Getrc3(w)
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.b ,
                Db7 = Portc.7 , Rs = Portd.7 , E = Portd.6
Config Lcd = 20 * 2
Dim W As Word

Do
   Call Getrc3(w)
   Cls
   Lcd W
   Waitms 300
Loop

Sub Getrc3(w)
Config Pind.3 = Output
   Portd.3 = 0
   Waitms 5
   W = 0
   Config Pind.3 = Input
   Portd.3 = 0
   Do
      If Pind.3 = 1 Then Exit Do
      Else
         W = W + 1
         If W > 65534 Then Exit Do
      End If
   Loop
End Sub
```

030036 - 17



the BAS routine is 10 times faster than the original MCS routine GetRC0, because with R_{max} = 10 kΩ and C = 100 nF a charging time of 5 ms (Waitms 5) is more than adequate.

Enter two port pins

The measurement will become a lot more accurate if the constant switching between input and output is avoided. This is made possible using the circuit in **Figure 2**, which 'occupies' two port pins. During the measurement, the capacitor is rapidly and fully charged via a low-value (470-Ω) resistor on port line d.4. Next, pin d.4 is configured as an input, pin d.3 as an output and pulled Low.

The program consists of a clever subroutine which takes the place of

W = GetRC9(pind,4)

in the first listing. The subroutine (**Listing 3**) pulls both port pins High and charges the capacitor in 10 ms. Next, pin d.4 is returned to 'input' and from then on watches the capacitor voltage. The capacitor is allowed to discharge all the time via port pin d.3. Software counter W is incremented on each loop iteration until the capacitor voltage has dropped below the Low limit. The loop is then left and the two port pins restored to the functions they had before the jump into the subroutine. Finally, there's the usual return instruction which takes the micro back to the main program.

The above measurement method enables errors of less than 1% to be achieved as long as the resistor value is within the range 1 kΩ to 100 kΩ. Obviously, sacrificing an extra port pin is well worth the effort!

If you are low on processor I/O pins already then a usable measurement may also be carried out using the circuit in **Figure 3**, but only if the resistor to be measured is not too low in value. The trick used in the routine $$Getrc3.bas (Listing 4) is that the capacitor is charged in the 'off' state and fully discharged via the port pin. The measurement sequence starts with port pin d.3 being configured as an input. The loop increments the counter until the voltage at the port pin reaches the High threshold value. Then follows the jump back to the main program.

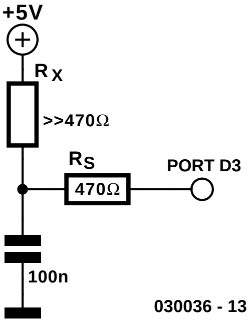


Figure 3. The port line saving alternative $$getrc3.bas.

(030036-1)